Machine learning project report

Tesla Stock Price Prediction
Time Series Forecasting

Team Members: [Bouleuc Martin, Seunghyun Kim]

December 9, 2025

Business scope

Context

The stock market is very hard to predict because it changes all the time. For traders, knowing if the price will go up or down is very important to make money. We chose Tesla (TSLA) because this stock is very volatile, so it is an interesting challenge for machine learning. The goal of this project is to see if we can predict the future price using past data.

Objective

We want to predict the close price of Tesla for the next day. To do this, we use historical data and some technical indicators. The idea is to have a model with a low error (RMSE) to help decide if we should buy or sell.

Data and methodology

Dataset

We used the library `yfinance` to download the data of Tesla from January 2020 to November 2025. We have 1,435 lines of data. The columns are: Open, High, Low, Close, and Volume. Our target variable is the close price, but shifted by -1 day (because we want to guess the next day).

Feature engineering

The raw price is not enough for the model. So, we calculated some financial indicators to give more context:

- SMA (simple moving averages): We took 20 and 50 days to see the trends.
- RSI: To see if the stock is overbought or oversold.
- MACD: To detect changes in the momentum.
- Bollinger bands: To see the volatility.

Exploratory data analysis (EDA)

First, we looked at the data to understand the links between variables.

We computed a correlation matrix. As we can see in Figure 1, the variables SMA_20 and SMA_50 are very correlated with the target. This is logical because the moving average follows the price. The RSI is less correlated but still useful.
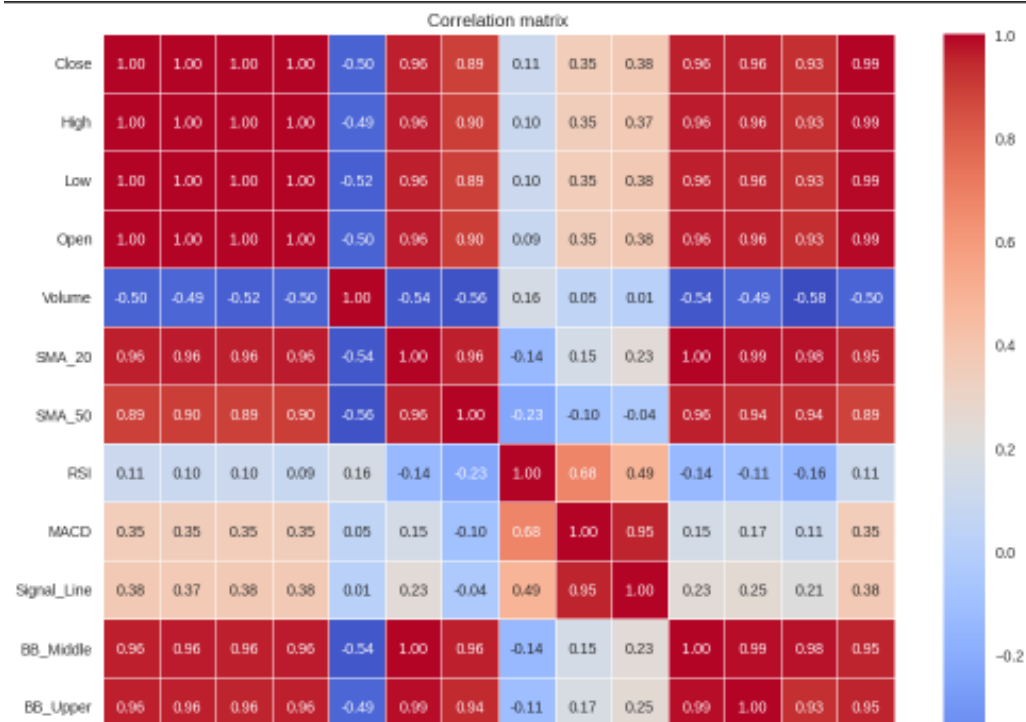
Figure 1: Correlation Matrix

We also tried to do a PCA to see if we can reduce the number of columns. The graph below (Figure 2) shows that we can keep 95% of the variance with fewer components.
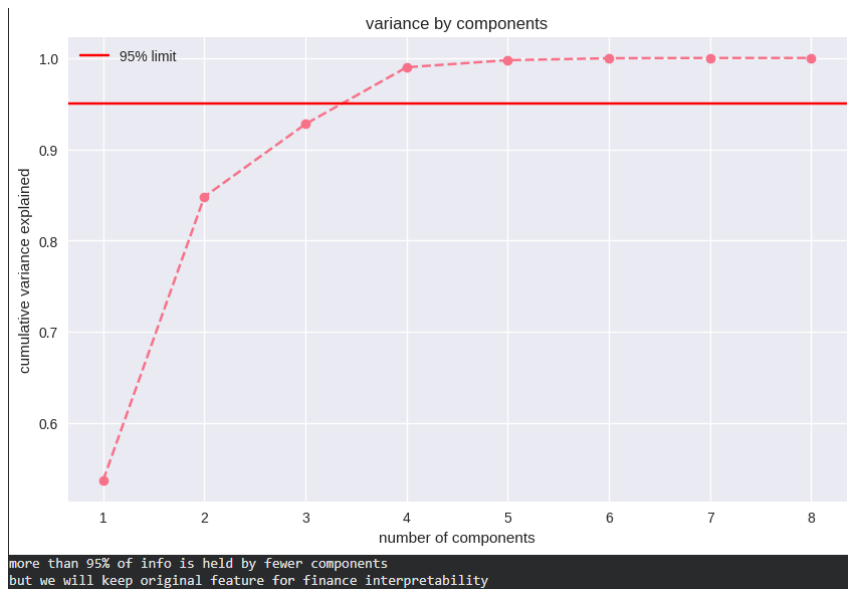

Figure 2: Explained Variance by PCA

However, we decided not to use the PCA components for the final model. We prefer to keep the original indicators (RSI, MACD) because it is easier to explain to a trader or a client.

Preprocessing

For the split, we did not use random shuffling because it is a time series. We used the first 80% of dates for training and the last 20% for testing. Also, for the LSTM model, we scaled the data between 0 and 1 using `MinMaxScaler`.

Models

We tested three different models to compare them.

Linear regression

We started with a simple linear regression to have a baseline. It assumes the relation is simple, which is probably not true for stocks, but it gives a first idea of the error.

XGBoost

Then we tried XGBoost. It is a very popular algorithm for tabular data [2]. We used `RandomizedSearchCV` to find good hyperparameters like the number of trees or the max depth. The problem with XGBoost is that it is not very good at predicting values outside the range of the training set (extrapolation).

LSTM

Finally, we built a recurrent neural network with LSTM (long short-term memory) [1]. This model is different because it has a "memory". We gave it a lookback window of 60 days. It means the model looks at the last 2 months to predict tomorrow. This is very adapted for time series.

Results

To compare the models, we use the RMSE (root mean squared error). We want this number to be as low as possible.

Comparison

In the graph below (Figure 3), we plot the real price (Black) and the predictions.

- The linear regression (Blue dashed) is too simple.
- The XGBoost (Green) is okay but sometimes it "stalls" when the price goes too high.
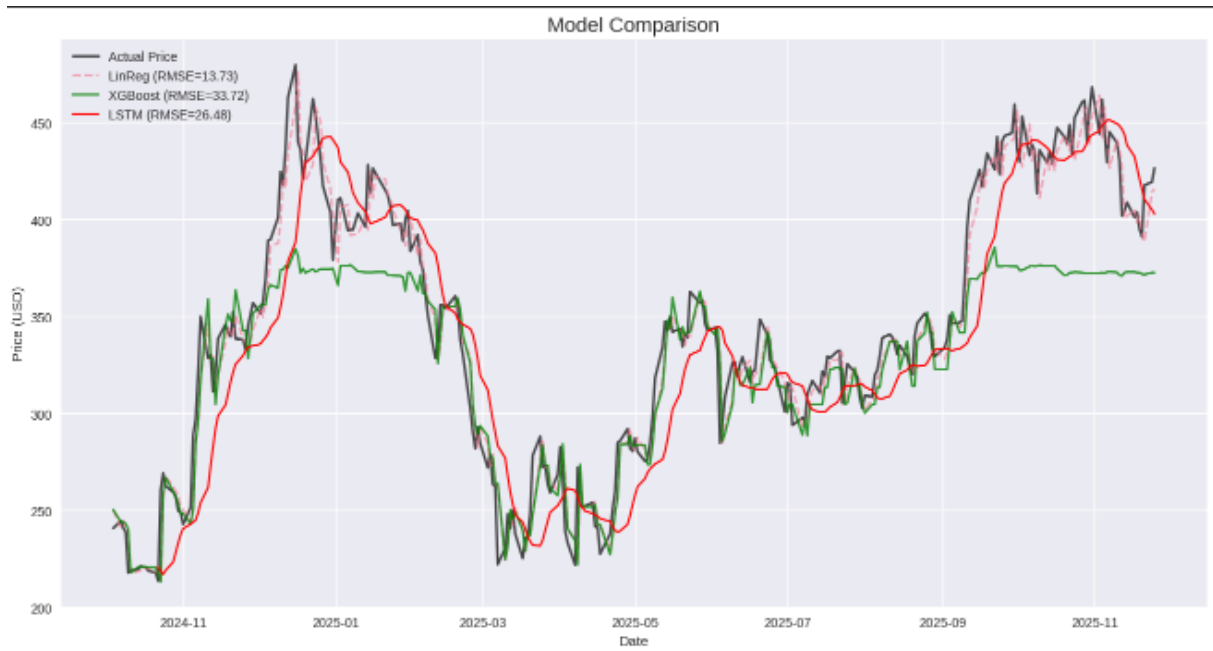- The LSTM (Red) is the best. It follows the variations of the price very well.

Figure 3: Predictions vs Real Price

Overfitting

Since financial data is noisy, we checked if the LSTM was overfitting. We plotted the loss curve during training.
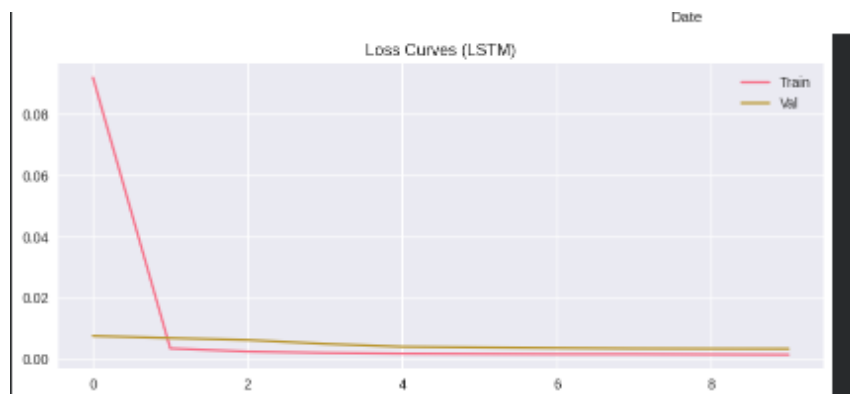

Figure 4: Train vs Validation Loss

We see that the validation loss decreases with the training loss, so the model is learning correctly without just memorizing the data.

Conclusion

Limitations

Even if the LSTM works well, there is a big limitation. Our model only looks at the price and technical indicators. It does not know about the news. For example, if Elon Musk

tweets something or if there is a new regulation, the price will change, but the model cannot predict it. This is called "fundamental analysis" and it is missing here.

Summary

In this project, we showed that deep learning (LSTM) is better than classical machine learning (linear, XGBoost) for stock prediction. The LSTM can capture the temporal aspect of the data thanks to the lookback window. Also, adding features like RSI helped the model.

More?

To improve this project, we could try to add sentiment analysis. We could scrape news titles, analyze if they are positive or negative, and give this score to the LSTM.

References

[1] Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*. Neural Computation.

[2] Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. KDD '16.

[3] Yahoo Finance API. *yfinance library documentation*. Online.